

webMethods EntireX

EntireX z/VSE Batch RPC Server

Version 9.6

April 2014

This document applies to webMethods EntireX Version 9.6.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1997-2014 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors..

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://documentation.softwareag.com/legal/>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://documentation.softwareag.com/legal/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices and license terms, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". This document is part of the product documentation, located at <http://documentation.softwareag.com/legal/> and/or in the root installation directory of the licensed product(s).

Document ID: EXX-BATCHRPC-VSE-96-20140628

Table of Contents

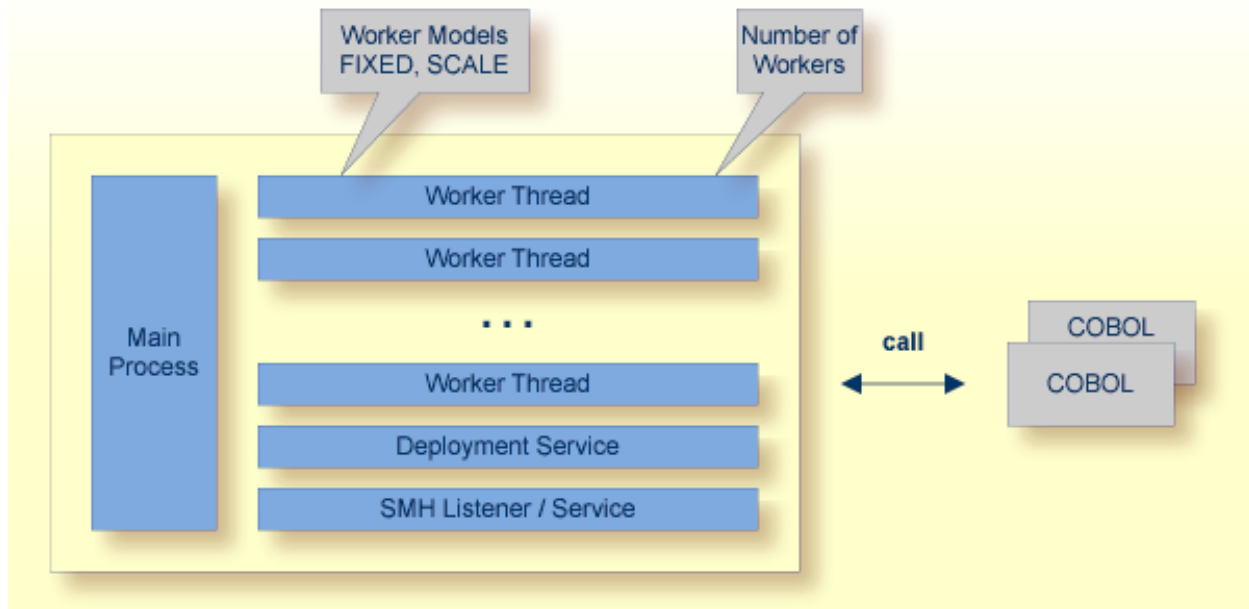
| | |
|--|----|
| 1 Introduction to the Batch RPC Server | 1 |
| Worker Models | 2 |
| Inbuilt Services | 3 |
| Usage of SVM Files | 4 |
| 2 Administering the Batch RPC Server | 7 |
| Customizing the RPC Server with a Configuration File | 8 |
| Configuring the RPC Server | 8 |
| Locating and Calling the Target Server | 15 |
| Starting the RPC Server | 16 |
| Stopping the RPC Server | 16 |
| Activating Tracing for the RPC Server | 16 |
| 3 Deployment Service | 17 |
| Introduction | 18 |
| Scope | 19 |
| Enabling the Deployment Service | 19 |
| Disabling the Deployment Service | 20 |
| 4 Handling SVM Files | 21 |
| SVM Files in the EntireX Workbench | 22 |
| SVM Files in the RPC Server | 22 |
| Source Control of SVM Files | 23 |
| Change Management of SVM Files | 23 |
| Compare SVM Files | 23 |
| List Deployed SVM Files | 23 |
| Check if an SVM File Revision has been Deployed | 24 |
| Access Control: Secure SVM File Deployment | 24 |
| Ensure that Deployed SVM Files are not Overwritten | 24 |
| When is an SVM File Required? | 24 |
| Is There a Way to Smoothly Introduce SVM Files? | 26 |
| 5 Scenarios | 27 |
| COBOL Scenarios | 28 |

1 Introduction to the Batch RPC Server

- Worker Models 2
- Inbuilt Services 3
- Usage of SVM Files 4

The EntireX z/VSE Batch RPC Server allows standard RPC clients to communicate with RPC servers on the operating system z/VSE under Batch. It supports the programming language COBOL and works together with the *COBOL Wrapper* and *IDL Extractor for COBOL*.

Worker Models



RPC requests are worked off inside the RPC server in worker threads, which are controlled by a main thread. Every RPC request occupies during its processing a worker thread. If you are using RPC conversations, each RPC conversation requires its own thread during the lifetime of the conversation. The Batch RPC Server provides two worker models:

- **FIXED**
The *fixed* model creates a fixed number of worker threads. The number of worker threads does not increase or decrease during the lifetime of an RPC server instance.
- **SCALE**
The *scale* model creates worker threads depending on the incoming load of RPC requests.

A maximum number (thru value of the `workermodel` parameter) of worker threads created can be set to restrict the system load. The minimum number (from value of the `workermodel` parameter), allows you to define a certain number of threads - not used by the currently executing RPC request - to wait for new RPC client requests to process. In this way the RPC server is ready to handle many RPC client requests arriving at the same time.

See parameter `workermodel` under *Configuring the RPC Server*.

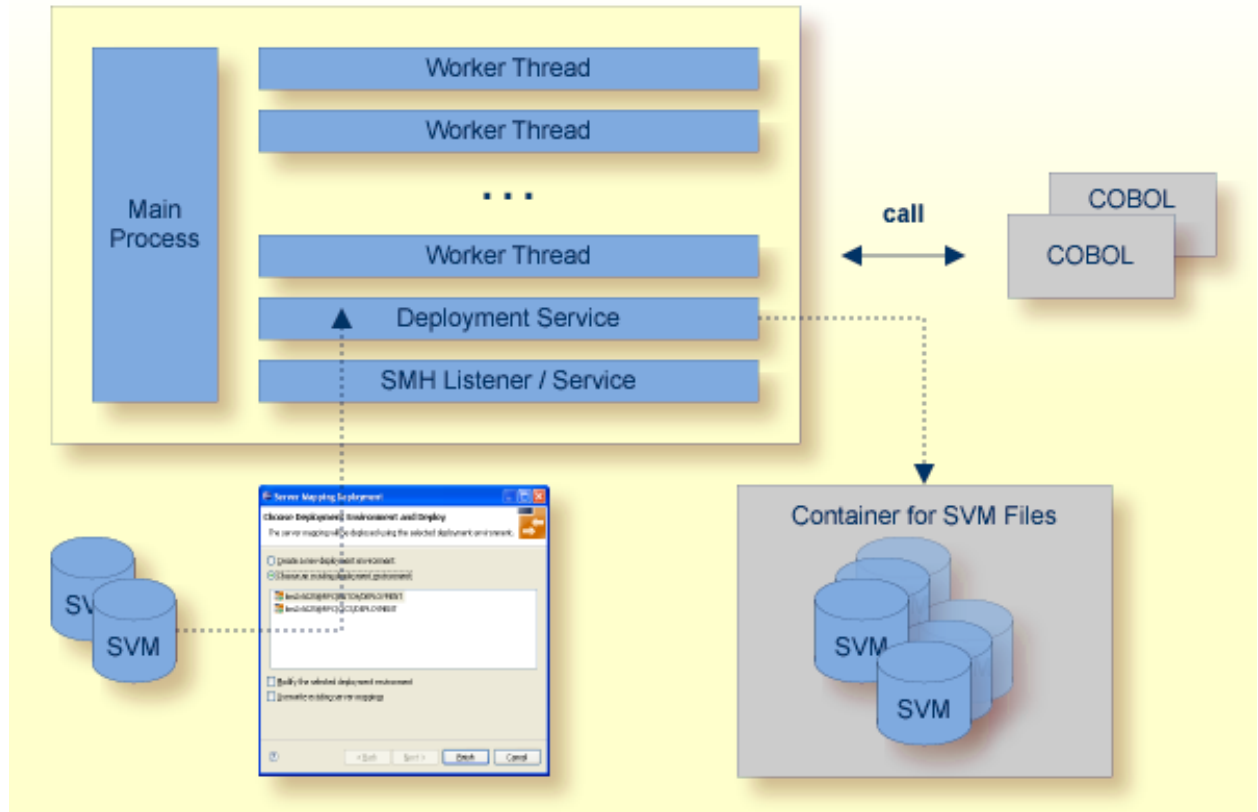
Inbuilt Services

The Batch RPC Server provides the following services for ease-of-use:

- [Deployment Service](#)
- [SMH Listener Service](#)

Deployment Service

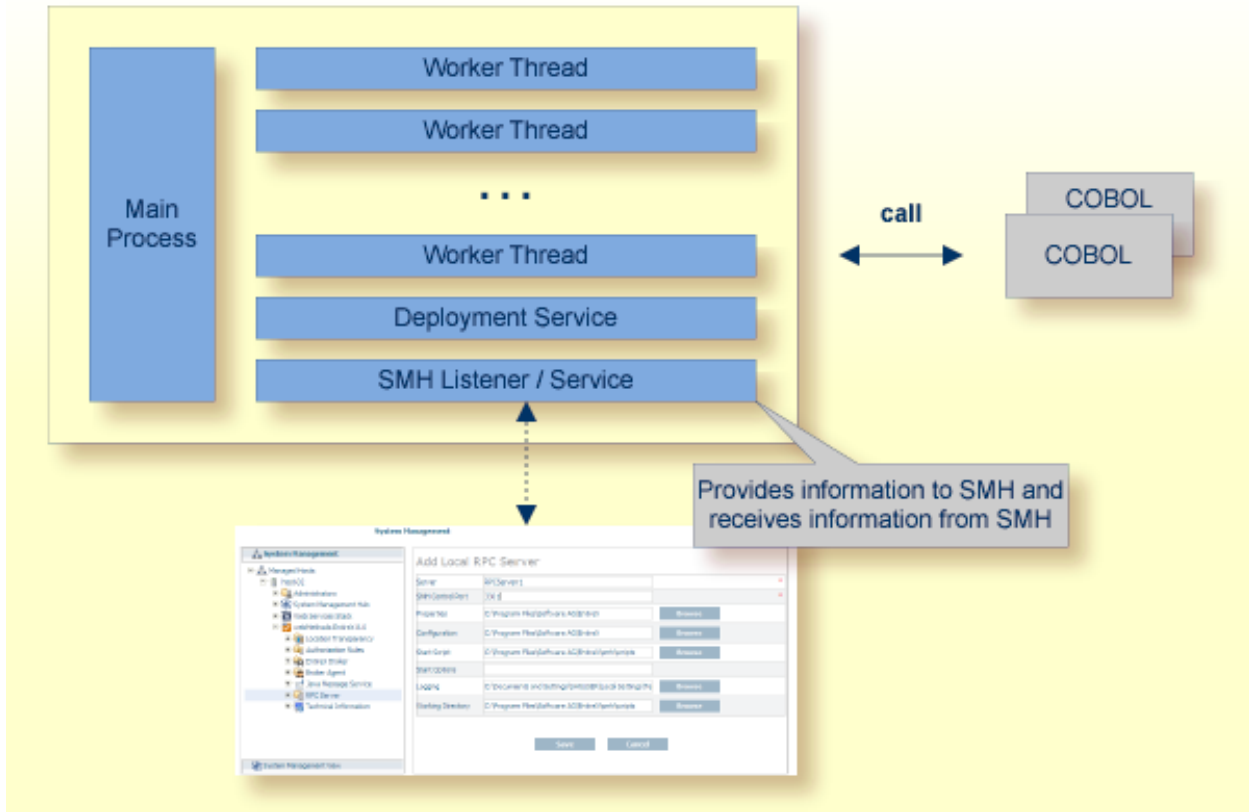
The Deployment Service allows you to deploy server mapping files (SVM files) interactively using the Deployment Wizard (see *Server Mapping Deployment*). On the RPC server side, the SVM files are stored in a VSAM file as the container. See [Deployment Service](#) for configuration information.



SMH Listener Service

With the SMH Listener Service you use the System Management Hub to monitor the RPC server. See *Administering the EntireX RPC Servers using System Management Hub* in the UNIX and Windows administration documentation.

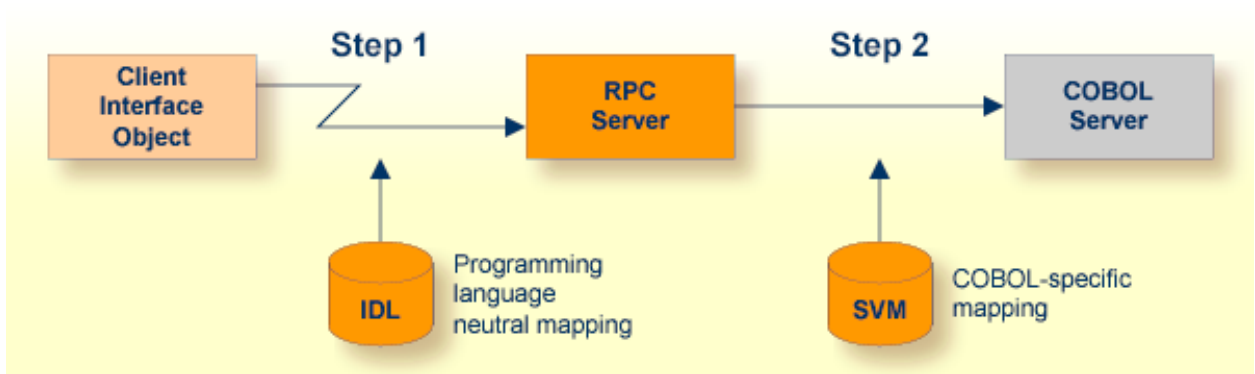
The SMH Service is switched on if the parameter `smhport` is set. See parameter `smhport` under *Configuring the RPC Server*.



Usage of SVM Files

There are many situations where the Batch RPC Server requires a server mapping file to correctly support special COBOL syntax such as JUSTIFIED, SYNCHRONIZE and OCCURS DEPENDING ON clauses, LEVEL-88 fields, etc. the .


SVM files contain COBOL-specific mapping information that is not included in the IDL file and therefore *not* sent by an EntireX RPC client to the RPC server. See also *When is an SVM File Required?* under *SVM Files*.



The RPC server marshalls the data in a two-step process: the RPC request coming from the RPC client (Step 1) is completed with COBOL-specific mapping information taken from the SVM file (Step 2). In this way the COBOL server can be called as expected.

The SVM files are retrieved as a result of the *IDL Extractor for COBOL* extraction process and the *COBOL Wrapper* if a COBOL server is generated.

You can customize the usage of the SVM file using parameter `svm`. See [Configuring the RPC Server](#).

 **Note:** SVM files are used for COBOL only.

2 Administering the Batch RPC Server

- Customizing the RPC Server with a Configuration File 8
- Configuring the RPC Server 8
- Locating and Calling the Target Server 15
- Starting the RPC Server 16
- Stopping the RPC Server 16
- Activating Tracing for the RPC Server 16

The EntireX z/VSE Batch RPC Server allows standard RPC clients to communicate with RPC servers on the operating system z/VSE under Batch. It supports the programming language COBOL and works together with the *COBOL Wrapper* and *IDL Extractor for COBOL*.

Customizing the RPC Server with a Configuration File

The name of the delivered example configuration file is `RPCPARM.CFG` (see sublibrary EXP960). The configuration file contains the configuration for the Batch RPC Server. The following settings are important:

- connection information such as broker ID, server address (class, name, service)
- location and usage of server mapping files (SVMs)
- scalability parameters
- trace settings
- etc.

For more information see [Configuring the RPC Server](#).

Configuring the RPC Server

The following rules apply:

- In the configuration file:
 - Comments must be on a separate line.
 - Comment lines can begin with `'*`, `'/` and `';`.
 - Empty lines are ignored.
 - Headings in square brackets [`<topic>`] are ignored.
 - Keywords are not case-sensitive.
- Underscored letters in a parameter indicate the minimum number of letters that can be used for an abbreviated command.

For example, in `brokerid=localhost`, `brok` is the minimum number of letters that can be used as an abbreviation, i.e. the commands/parameters `broker=localhost` and `brok=localhost` are equivalents.

| Parameter | Default | Values | Req/ Opt |
|----------------------|-----------|---|----------|
| <u>brokerid</u> | localhost | <p>Broker ID used by the server. See <i>Using the Broker ID in Applications</i> in the RPC Programming documentation.</p> <p>Example: brokerid=myhost.com:1971</p> | R |
| <u>class</u> | RPC | <p>Server class part of the server address used by the server. The server address must be defined as a service in the broker attribute file (see <i>Service-specific Attributes (DEFAULTS=SERVICE)</i> under <i>Broker Attributes</i> in the platform-independent administration documentation). Case-sensitive, up to 32 characters. Corresponds to CLASS.</p> <p>Example: class=MyRPC</p> | R |
| <u>codepage</u> | | <p>Depending on the internationalization approach, the codepage (locale string) where incoming data is provided to the COBOL server. Conversely, the COBOL server must provide outgoing data in the given codepage, otherwise unpredictable results occur. See <i>What is the Best Internationalization Approach to use?</i> under <i>Internationalization with EntireX</i> for information on which internationalization approach requires a codepage (locale string).</p> <p>By default, no codepage is transferred to the broker. For the most popular internationalization approach, <i>ICU Conversion</i> under <i>Introduction to Internationalization</i>, the correct codepage (locale string) must be provided. This means it must:</p> <ul style="list-style-type: none"> ■ follow the rules described under <i>Locale String Mapping</i> in the internationalization documentation ■ be a codepage supported by the broker ■ be the codepage used in your environment for file and terminal IO, otherwise unpredictable results may occur. <p>Example: codepage=ibm-273</p> | |
| <u>compresslevel</u> | N | <p>Enforce compression when data is transferred between broker and server. See <i>Data Compression in EntireX Broker</i> in the general administration documentation.</p> <p>compresslevel= 0 1 2 3 4 5 6 7 8 9 Y N</p> <p>0-9 0=no compression 9=max. compression</p> <p>N No compression.</p> | O |

| Parameter | Default | Values | Req/Opt |
|------------------------|---------|---|---------|
| | | <p>Y Compression level 6.</p> <p>Example: compresslevel=6</p> | |
| <u>deployment</u> | NO | <p>Activates the deployment service, see <i>Deployment Service</i>. Required to use the deployment wizard. See <i>Server Mapping Deployment Wizard</i> in the COBOL Wrapper documentation.</p> <p>YES Activates the deployment service. The RPC server registers the deployment service in the broker.</p> <p>NO The deployment service is deactivated. The RPC server does not register the deployment service in the broker.</p> <p>Example: deployment=yes</p> | O |
| <u>encryptionlevel</u> | 0 | <p>Enforce encryption when data is transferred between client and server. Requires EntireX Security. See ENCRYPTION-LEVEL under <i>Broker ACI Fields</i>.</p> <p>0 Encryption is enforced.</p> <p>1 Encryption is enforced between server and broker kernel.</p> <p>2 Encryption is enforced between server and broker kernel, and also between client and broker.</p> <p>Example: encryptionlevel=2</p> | O |
| <u>etblnk</u> | BKIMB | <p>Define the broker stub to be used. See <i>Administration of Broker Stubs under z/VSE</i> for available stubs.</p> <p>Example: ETBL=BKIMB</p> | O |
| <u>logon</u> | YES | <p>Execute broker functions LOGON/LOGOFF in worker threads. Must match the setting of the broker attribute AUTOLOGON. Reliable RPC requires logon set to YES. See <i>Reliable RPC</i>.</p> <p>NO No logon/logoff functions are executed.</p> <p>YES Logon/logoff functions are executed.</p> <p>Example: logon=no</p> | O |

| Parameter | Default | Values | Req/Opt | | | | |
|----------------------|--|--|---------|--|---|---|---|
| <u>marshalling</u> | COBOL | <p>The Batch RPC Server can be configured to support either COBOL or C. See also <i>Locating and Calling the Target Server</i>.</p> <p>marshalling=(LANGUAGE=COBOL C)</p> <table border="1"> <tr> <td>COBOL</td> <td>Server supports COBOL. The COBOL servers are called directly without a server interface object. So-called server mapping (SVM) files are used to call the COBOL server correctly if one is available. See <i>Server Mapping Deployment</i>.</td> </tr> <tr> <td>C</td> <td>Server supports C. The modules are called using a server interface object built with the <i>C Wrapper</i>.</td> </tr> </table> | COBOL | Server supports COBOL. The COBOL servers are called directly without a server interface object. So-called server mapping (SVM) files are used to call the COBOL server correctly if one is available. See <i>Server Mapping Deployment</i> . | C | Server supports C. The modules are called using a server interface object built with the <i>C Wrapper</i> . | O |
| COBOL | Server supports COBOL. The COBOL servers are called directly without a server interface object. So-called server mapping (SVM) files are used to call the COBOL server correctly if one is available. See <i>Server Mapping Deployment</i> . | | | | | | |
| C | Server supports C. The modules are called using a server interface object built with the <i>C Wrapper</i> . | | | | | | |
| <u>password</u> | no default | <p>Password for broker logon. Case-sensitive, up to 32 characters. For more information see broker ACI control block field PASSWORD.</p> <p>Example: password=MyPwd</p> | O | | | | |
| <u>restartcycles</u> | 15 | <p>Number of restart attempts if the broker is not available. This can be used to keep the Batch RPC Server running while the broker is down for a short time. A restart cycle will be repeated at an interval which is calculated as follows:</p> <p>timeout + ETB_TIMEOUT + 60 seconds</p> <p>where timeout is the RPC server parameter (see this table), and</p> <p>ETB_TIMEOUT is the environment variable (see <i>Environment Variables in EntireX</i> in the general administration documentation)</p> <p>When the number of cycles is reached and a connection to the broker is not possible, the RPC server stops.</p> <p>Example: restartcycles=30</p> | O | | | | |
| <u>runoption</u> | no default | <p>This parameter is for special purposes. It provides the Batch RPC Server with additional information. The runoptions are normally set to meet the platform's requirements. Set this parameter only if a support representative provides you with an option and asks you to do so. The parameter can be defined multiple times.</p> | O | | | | |

| Parameter | Default | Values | Req/ Opt |
|-------------------|---------|---|----------|
| | | <p>Example: runoption=<option> runoption=<option></p> | |
| <u>servername</u> | SRV1 | <p>Server name part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See <i>Service-specific Attributes</i> (DEFAULTS=SERVICE) under <i>Broker Attributes</i> in the platform-independent administration documentation. Case-sensitive, up to 32 characters. Corresponds to SERVER of the broker attribute file.</p> <p>Example: servername=mySrv</p> | R |
| <u>service</u> | CALLNAT | <p>Service part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See <i>Service-specific Attributes</i> (DEFAULTS=SERVICE) under <i>Broker Attributes</i> in the platform-independent administration documentation. Case-sensitive, up to 32 characters. Corresponds to SERVICE attribute of the broker attribute file.</p> <p>Example: service=MYSERVICE</p> | R |
| <u>smhport</u> | 0 | <p>The port where the server listens for commands from the System Management Hub (SMH). If this port is 0 (default), no port is used and management by the SMH is disabled.</p> <p>Example: smhport=3001</p> | O |
| <u>svm</u> | ERXSVM | <p>Usage and location of SVM files. If no <code>svm</code> parameter is given, the RPC server tries to open the SVM container using DLBL name ERXSVM. If this DLBL name is not available, no server mappings are used. For more information see SVM Files.</p> <pre>svm = no d1blname</pre> <p>no No SVM files are used. d1blname DLBL name of the SVM file container in the startup JCL of the Batch RPC Server.</p> <p>Example: svm=MY SVM</p> <p>For the example above, define the DLBL name MY SVM in the startup JCL of the Batch RPC Server as</p> | O |

| Parameter | Default | Values | Req/ Opt |
|--------------------------|---------|--|----------|
| | | <pre>// DLBL ↵ MYSVM, 'ENTIREX.SVMDEV.KSDS', 0, VSAM, CAT=VSESPUC</pre> <p>See Step 3: Customize the Batch RPC Server Startup JCL - RUNRPC.J under Installing the z/VSE EntireX RPC Servers.</p> | |
| <code>timeout</code> | 60 | <p>Timeout in seconds, used by the server to wait for broker requests. See broker ACI control block field WAIT for more information. Also influences <code>restartcycles</code>.</p> <p>Example: <code>timeout=300</code></p> | O |
| <code>tracelevel</code> | None | <p>Trace level for the server. See also Activating Tracing for the RPC Server.</p> <pre>tracelevel = None Standard Advanced ↵ Support</pre> <p>None No trace output. Standard For minimal trace output. Advanced For detailed trace output. Support This trace level is for support diagnostics and should only be switched on when requested by Software AG support.</p> <p>Example: <code>tracelevel=standard</code></p> | O |
| <code>traceoption</code> | None | <p>Additional trace option if trace is active.</p> <p>None No additional trace options. STUBLOG If <code>tracelevel</code> is Advanced or Support, the trace additionally activates the broker stub log. NOTRUNC Normally if a data buffer larger than 8 KB is traced, the buffer trace is truncated. Set this option to write the full amount of data without truncation.</p> <p>Note: This can increase the amount of trace output data dramatically if you transfer large data buffers.</p> <p>Example: <code>traceoption=(STUBLOG,NOTRUNC)</code></p> | O |

| Parameter | Default | Values | Req/ Opt | | | | | | | | |
|--------------------------|--|--|----------|--|-------|--|------------|--|------------|---|---|
| <code>userid</code> | ERX-SRV | <p>Used to identify the server to the broker. See broker ACI control block field <code>USER-ID</code>. Case-sensitive, up to 32 characters.</p> <p>Example: <code>userid=MyUid</code></p> | R | | | | | | | | |
| <code>workermodel</code> | SCALE,1,3,slowshrink | <p>The Batch RPC Server can be configured to</p> <ul style="list-style-type: none"> adjust the number of worker threads to the current number of client requests: <pre>workermodel=(SCALE,from,thru [,slowshrink fastshrink])</pre> use a fixed number of worker threads: <pre>workermodel=(FIXED,number)</pre> <table border="1"> <tr> <td>FIXED</td> <td>A fixed <i>number</i> of worker threads is used by the Batch RPC Server.</td> </tr> <tr> <td>SCALE</td> <td>The number of worker threads is adjusted to the current number of client requests. With the <i>from</i> value, the minimum number of active worker threads can be set. The <i>thru</i> value restricts the maximum number of worker threads.</td> </tr> <tr> <td>slowshrink</td> <td>The RPC server stops all worker threads not used in the time specified by the <code>timeout</code> parameter, except for the number of workers specified as minimum value. This is the default if SCALE is used.</td> </tr> <tr> <td>fastshrink</td> <td>The RPC server stops worker threads immediately as soon as it has finished its conversation, except for the number of workers specified as minimum value.</td> </tr> </table> <p>Example: <code>workermodel=(SCALE,2,5)</code></p> | FIXED | A fixed <i>number</i> of worker threads is used by the Batch RPC Server. | SCALE | The number of worker threads is adjusted to the current number of client requests. With the <i>from</i> value, the minimum number of active worker threads can be set. The <i>thru</i> value restricts the maximum number of worker threads. | slowshrink | The RPC server stops all worker threads not used in the time specified by the <code>timeout</code> parameter, except for the number of workers specified as minimum value. This is the default if SCALE is used. | fastshrink | The RPC server stops worker threads immediately as soon as it has finished its conversation, except for the number of workers specified as minimum value. | O |
| FIXED | A fixed <i>number</i> of worker threads is used by the Batch RPC Server. | | | | | | | | | | |
| SCALE | The number of worker threads is adjusted to the current number of client requests. With the <i>from</i> value, the minimum number of active worker threads can be set. The <i>thru</i> value restricts the maximum number of worker threads. | | | | | | | | | | |
| slowshrink | The RPC server stops all worker threads not used in the time specified by the <code>timeout</code> parameter, except for the number of workers specified as minimum value. This is the default if SCALE is used. | | | | | | | | | | |
| fastshrink | The RPC server stops worker threads immediately as soon as it has finished its conversation, except for the number of workers specified as minimum value. | | | | | | | | | | |

Locating and Calling the Target Server

The approach used to derive the z/VSE module name for the RPC server depends on whether so-called server mapping files are used or not. See [Usage of SVM Files](#) for an introduction. This section applies to COBOL.

- If SVM files are used, the IDL library and IDL program names are used to form a key to locate the SVM entry in the SVM container. If an SVM entry is found, the z/VSE module name of the RPC server is derived from the SVM entry. In this case the IDL program name can be different to the z/VSE module name if it is renamed during wrapping process (see *Customize Automatically Generated Server Names*) or during the extraction process in the COBOL Mapping Editor (see *The Software AG IDL Tree Pane*).
- If no SVM files are used at all, the IDL program name is used as the z/VSE module name of the RPC server (the IDL library name is ignored).

▶ To use the Batch RPC Server with COBOL

- 1 Make sure that all z/VSE modules called as RPC servers
 - are compiled with IBM's Language Environment (see [LE/VSE V1R4 Programming Guide](#) for more information)
 - use COBOL calling conventions
 - can be called dynamically ("fetched") from any Language Environment program
 - are accessible through the Batch RPC Server JCL LIBDEF chain.

- 2 Configure the parameter `marshalling` for COBOL, for example:

```
marshalling=COBOL
```

- 3 Configure the parameter `svm` depending on whether SVM files are used or not.

See also [Scenario I: Calling an Existing COBOL Server](#) or [Scenario II: Writing a New COBOL Server](#).

Starting the RPC Server

▶ To start the Batch RPC Server

- Run the job `RPCRPC.J`.

Stopping the RPC Server

▶ To stop the Batch RPC Server

- Use the console command `STOP`. For example:

```
task_id STOP
```

Or:

Use the System Management Hub. This method ensures that the deregistration from the Broker is correct.

Activating Tracing for the RPC Server

▶ To activate tracing for the Batch RPC Server

- 1 Set the parameter `tracelevel`.
- 2 Dynamically change the trace level with the operator command

```
port_number TRACELEVEL=tracelevel
```

See the table below for supported trace levels.

The `TRACELEVEL` command without `tracelevel` option will report the currently active trace.

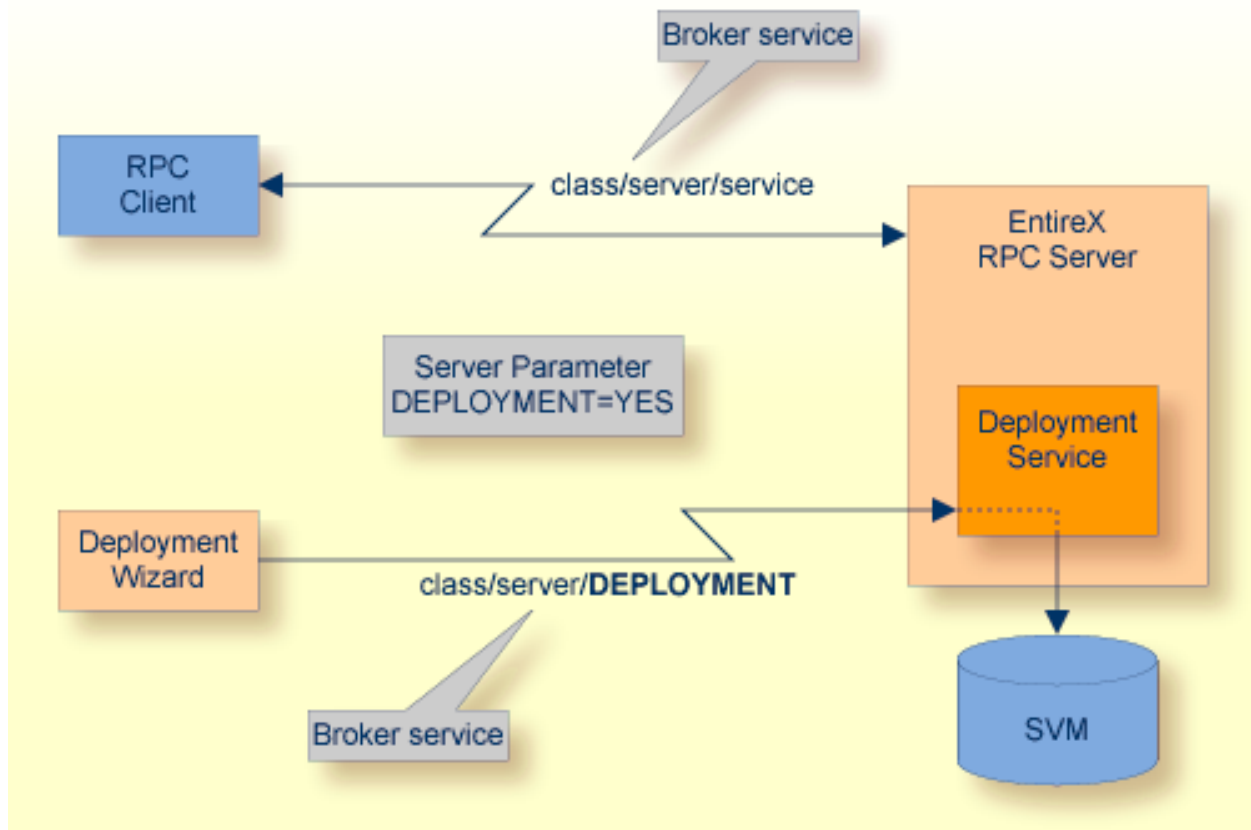
3 Deployment Service

- Introduction 18
- Scope 19
- Enabling the Deployment Service 19
- Disabling the Deployment Service 20

Introduction

The deployment service

- is the (server-side) counterpart to the deployment wizard; see *Server Mapping Deployment Wizard*.
- is a built-in service of the EntireX RPC server, which can be enabled/disabled by EntireX RPC server configuration settings
- usage can be restricted to certain users or group of users, using EntireX Security; see *Authorization of Client and Server* under *Overview of EntireX Security* in the EntireX Security documentation.



Scope

The deployment service is used for the

- IDL Extractor for COBOL to deploy SVM files with the deployment wizard;
- COBOL Wrapper for server generation to deploy SVM files with the deployment wizard.

See *Server Mapping Deployment Wizard*.

The deployment service uses the same class and server names as defined for the EntireX RPC server, and DEPLOYMENT as the service name, resulting in `class/server/DEPLOYMENT` as the broker service. Please note DEPLOYMENT is a service name reserved by Software AG. See broker attribute SERVICE.

Enabling the Deployment Service

▶ To enable the deployment service

- 1 For a Batch RPC Server, the server mapping file VSAM (container) must be installed and configured. See *Step 1: Define an RPC Server mapping file (SVM File) - VSAMDEF.J (Optional)* under *Installing the Batch RPC Server* under *Installing the z/VSE EntireX RPC Servers*.
- 2 Set the RPC server parameter `deployment=yes`. See `deployment` under *Configuring the RPC Server*.
- 3 Define in the broker attribute file, under the RPC service, an additional broker service with DEPLOYMENT as the service name and values for class and server identical to those used for the RPC service. For example, if your RPC service is named

```
CLASS = RPC    SERVER = SRV1    SERVICE = CALLNAT
```

the deployment service requires the following additional service definition in the broker attribute file:

```
CLASS = RPC    SERVER = SRV1    SERVICE = DEPLOYMENT
```

- 4 Optional. If you need to restrict the use of the deployment service to a selected group of users, use EntireX Security and define security rules for the `class/server/DEPLOYMENT` broker service. The service name DEPLOYMENT is a constant.
 - For a z/OS broker, see *Resource Profiles in EntireX Security* in the EntireX Security documentation.

- For a UNIX or Windows broker, see *Administering Authorization Rules using System Management Hub* in the UNIX and Windows administration documentation.
- Not applicable to a BS2000/OSD or z/VSE broker.

Disabling the Deployment Service

▶ To disable the deployment service

- Set the Batch RPC Server parameter `deployment=no`. See `deployment` under *Configuring the RPC Server*.

The Batch RPC Server will not register the deployment service in the broker.

4 Handling SVM Files

- SVM Files in the EntireX Workbench 22
- SVM Files in the RPC Server 22
- Source Control of SVM Files 23
- Change Management of SVM Files 23
- Compare SVM Files 23
- List Deployed SVM Files 23
- Check if an SVM File Revision has been Deployed 24
- Access Control: Secure SVM File Deployment 24
- Ensure that Deployed SVM Files are not Overwritten 24
- When is an SVM File Required? 24
- Is There a Way to Smoothly Introduce SVM Files? 26

A server mapping file (SVM) enables the RPC server to correctly support special COBOL syntax such as `REDEFINES`, `JUSTIFIED`, `SYNCHRONIZE` and `OCCURS DEPENDING ON` clauses, `LEVEL-88` fields, etc. If one of these elements is used, the EntireX Workbench automatically extracts an SVM file in addition to the IDL (interface definition language), or an SVM file is generated by the COBOL Wrapper for a server skeleton. The SVM file is used at runtime to marshal and unmarshal the RPC data stream.

SVM Files in the EntireX Workbench

In the *EntireX Workbench*, an SVM file has to relate to an appropriate IDL file. Therefore, you always have to keep the IDL file and the SVM file together in the same folder.

If there is an SVM file and a corresponding IDL file,

- at least one of the IDL programs in the corresponding IDL file requires server-mapping information to correctly call the target server. For those IDL programs, there is an SVM entry (line) in the Workbench SVM file.
- deployment of the SVM file to the RPC server is mandatory, see *Server Mapping Deployment*.

If there is an IDL file but no corresponding SVM file,

- there is no IDL program that requires server mapping information.

SVM Files in the RPC Server

Under *z/VSE*, SVM entries of the EntireX Workbench SVM files are stored as records within one VSAM file (containing all SVM entries from all Workbench SVM files). The unique key of the VSAM file consists of the first 255 bytes of the record: for the type (1 byte), for the IDL library (127 bytes) and for the IDL program (127 bytes). The CICS, Batch and IMS RPC servers use a VSAM file as the container.

If *one* server requires an SVM file, you need to provide this to the RPC server:

- Development environments: to allow the deployment of new SVM files, enable the deployment service. See *Enabling the Deployment Service*.
- Production environments: provide SVM files to the RPC server.

If *no* server requires an SVM file, you can execute the RPC server without SVM files:

- Development environments: you can disable the deployment service. See *Disabling the Deployment Service*.
- Production environments: there is no need to provide SVM files to the RPC server. .

Source Control of SVM Files

Because SVM entries within an SVM file contain text data only, a Workbench SVM file is text-based (although it is not intended for human consumption). Therefore, you can include it in your source control management together with the IDL file and the COBOL source(s) as a triplet that should always be kept in sync.

Change Management of SVM Files

Under z/VSE, change management for a VSAM file (SVM container) is similar to change management for a database. The complete VSAM file can be backed up at any time, for example by using IDCAMS. All updates to the VSAM file done after a backup must be kept.

All Workbench SVM files added since the last backup should be available.

Compare SVM Files

For SVM files in the *EntireX Workbench* format, you can use a third party file/text compare tool to check if two files are identical.

The SVM entries (corresponding to lines in a Workbench SVM file) contain a creation timestamp at offset 276 (decimal) in the format *YYYYMMDDHHIISST*. The precision is 1/10 of a second.

List Deployed SVM Files

Use IDCAMS:

```
* $$ JOB JNM=VSAMPRNT,CLASS=0,DISP=D
* $$ LST CLASS=A,DISP=K
/* ----- */
/* PRINT CONTENT OF AN SVM VSAM CLUSTER */
/* ----- */
// JOB VSAMPRNT
// DLBL ERXSVM, 'ENTIREX.SVMDEV.KSDS',0,VSAM,CAT=VSESPUC
// EXEC IDCAMS,SIZE=AUTO
// PRINT INFILE(ERXSVM) CHAR
/*
/&
* $$ EOJ
```

Check if an SVM File Revision has been Deployed

SVM entries (corresponding to lines in Workbench SVM files) contain a creation timestamp at offset 276 (decimal) in the format *YYYYMMDDHHIISSST*. Precision is 1/10 of a second. The creation timestamp can be checked.

The timestamp can be found on the same offset in the records in the VSAM file (SVM container).

Access Control: Secure SVM File Deployment

For deployment with the *Server Mapping Deployment Wizard*, use EntireX Security if the broker is running on platforms z/OS, UNIX, Windows or z/VSE. See [Enabling the Deployment Service](#).

For IBM deployment tool IDCAMS, use RACF to secure deployment.

Ensure that Deployed SVM Files are not Overwritten

For IDCAMS, use the `NOREPLACE` option to disallow overwriting of duplicate SVM records in the VSAM file (container). See *Server Mapping Deployment using FTP and IDCAMS*.

When is an SVM File Required?

For the IDL Extractor for COBOL

| Interface Type | COBOL Syntax | COBOL Mapping Editor | SVM Required | More Information |
|--|--------------|----------------------|--------------|--|
| CICS with DFHCOMMAREA Calling Convention and IN different to OUT | all | | yes | <i>CICS with DFHCOMMAREA Calling Convention under Introduction to the IDL Extractor for COBOL CICS DFHCOMMAREA under COBOL Parameter Selection</i> |
| CICS Channel Container Calling Convention | all | | yes | <i>CICS with Channel Container Calling Convention</i> |
| CICS with DFHCOMMAREA Large Buffer Interface | all | | yes | <i>CICS with DFHCOMMAREA Large Buffer Interface</i> |

| Interface Type | COBOL Syntax | COBOL Mapping Editor | SVM Required | More Information |
|--|----------------------------------|----------------------|--------------|--|
| IMS MPP Message Interface (IMS Connect) | all | | yes | <i>IMS MPP Message Interface (IMS Connect)</i> |
| IMS BMP with Standard Linkage Calling Convention | all | | yes | <i>IMS BMP with Standard Linkage Calling Convention</i> |
| Micro Focus with Standard Linkage Calling Convention | BINARY clause | | yes | <i>Micro Focus with Standard Linkage Calling Convention</i> |
| all | OCCURS DEPENDING ON clause | | yes | <i>Tables with Variable Size - DEPENDING ON Clause under COBOL to IDL Mapping in the IDL Extractor for COBOL documentation</i> |
| all | REDEFINES clause | | yes | <i>REDEFINE Clause</i> |
| all | TRAILING [SEPARATE] clause | | yes | <i>SIGN LEADING and TRAILING SEPARATE Clause</i> |
| all | LEADING [SEPARATE] clause | | yes | <i>SIGN LEADING and TRAILING SEPARATE Clause</i> |
| all | ALIGNED RIGHT attribute | | yes | |
| all | all | Rename of program | yes | <i>The Software AG IDL Tree Pane under Mapping Editor User Interface in the IDL Extractor for COBOL documentation</i> |
| all | all | Map to operation | yes | <i>Context Menu under The COBOL Parameters Pane</i> |
| all | all | Map to constant | yes | <i>Context Menu</i> |
| all | all | Suppress | yes | <i>Context Menu</i> |
| other combinations | | | no | |

For the COBOL Wrapper

This depends on the interface type chosen and the IDL type:

| Interface Type | IDL Type | COBOL Wrapper | SVM Required | More Information |
|--|---------------------|--|--------------|---|
| CICS with DFHCOMMAREA Large Buffer Interface | all | | yes | <i>CICS with DFHCOMMAREA Large Buffer Interface under COBOL Server Interface Types</i> |
| CICS with Channel Container Calling Convention | all | | yes | <i>CICS with Channel Container Calling Convention</i> |
| IMS BMP with Standard Linkage Calling Convention | all | | yes | <i>IMS BMP with Standard Linkage Calling Convention</i> |
| Micro Focus | I2 or I4 | | yes | <i>Micro Focus with Standard Linkage Calling Convention IDL Data Types under Software AG IDL File in the IDL Editor documentation</i> |
| all | IDL unbounded array | | yes | <i>array-definition under Software AG IDL Grammar in the IDL Editor documentation</i> |
| all | IDL unbounded group | | yes | <i>group-parameter-definition under Software AG IDL Grammar in the IDL Editor documentation</i> |
| all | all | IDL program name is not a valid COBOL name and is therefore adapted, or the COBOL program name is customized | yes | <i>Customize Automatically Generated Server Names</i> |
| other combinations | | | no | |

Is There a Way to Smoothly Introduce SVM Files?

All EntireX RPC servers can be executed without SVM files. There is no need to install the SVM container (see [SVM Files in the RPC Server](#)) as long as you do not use features that require SVM files (see [When is an SVM File Required?](#)). You can also call COBOL servers generated or extracted with previous versions of EntireX mixed with a COBOL server that requires SVM files. All EntireX RPC servers are backward compatible.

5 Scenarios

- COBOL Scenarios 28

COBOL Scenarios

Scenario I: Calling an Existing COBOL Server

▶ **To call an existing COBOL server**

- 1 Use the *IDL Extractor for COBOL* to extract the Software AG IDL and, depending on the complexity of the extraction, also an SVM file.
- 2 Build an EntireX RPC client using any EntireX wrapper. See *EntireX Wrappers*. For a quick test you can:
 - use the IDL Tester; see *EntireX IDL Tester* in the EntireX Workbench documentation
 - generate an XML mapping file (XMM) and use the XML Tester for verification; see *EntireX XML Tester*

See *Client and Server Examples for z/VSE Batch* for COBOL RPC Server examples.

Scenario II: Writing a New COBOL Server

▶ **To write a new COBOL server**

- 1 Use the *COBOL Wrapper* to generate a COBOL server skeleton and, depending on the complexity of the extraction, also an SVM file. Write your COBOL server and proceed as described under *Using the COBOL Wrapper for the Server Side*.
- 2 Build an EntireX RPC client using any EntireX wrapper. See *EntireX Wrappers*. For a quick test you can:
 - use the IDL Tester; see *EntireX IDL Tester* in the EntireX Workbench documentation
 - generate an XML mapping file (XMM) and use the XML Tester for verification; see *EntireX XML Tester*

See *Client and Server Examples for z/VSE Batch* for COBOL RPC Server examples.